~~Method For Overload Protection For an Exchange~~

Automatic Congestion control (ACC) is a network-wide protection mechanism with which exchanges are protected from overload congestion. ACC is becoming increasingly important compared to other standardized approaches, since proprietary methods such as Siemens' delayed release are being restricted ~~in future by~~ by developing standards. In addition, ACC is the only protection mechanism that is provided for broadband exchanges. But the effectiveness of ACC is insufficient, as has been shown by network simulations. The performance of an exchange ~~slumps~~ decreases dramatically in an overload situation ~~. There is~~ and there is a great danger of buffer overruns.

The standard mechanism is described below.

Automatic Congestion Control Under the ITU Standard

Automatic Congestion Control is described in the ITU Standards E.412, E.542, Q.763 and Q.764 ~~. The basic~~ in which the basic idea is to choke off traffic in the direction of a congested exchange (VST) at the neighboring exchange ~~already; that is, not even to let it approach~~ before it arrives the congested VST. This approach entails three fundamental problems, on whose solution the quality of the algorithm depends:

- the information of the neighboring nodes,
- the control of the reactions at the neighboring nodes,
- the feedback via the success of the measures.

ACC provides 2 values to described congestion:

- An Automatic Congestion Level (ACL) of 1 signifies congestion,
- An ACL of 2 ~~stands for~~ signifies heavy congestion.

The ACL is determined at the congested VST ~~. It~~ and is transferred to the neighboring exchanges with the #7 message (release) when a call is initiated between the relevant exchanges (Figure 1). If a VST receives a positive ACL from a neighboring VST, a timer, ~~currently of~~ e.g., 10s, is wound up with the intention of providing that potential protection measures are stopped (i.e. canceled) again if a new positive ACL has not arrived after the timer runs out. ~~[sic]~~

At the neighboring VSTn of the congested VST, depending on the ACL value a certain percentage of all calls are denied or routed on an alternative path. The decision about which measure to take is up to the operator. The only rule is that the percentage must be a multiple of 12.5% (1/8).

Automatic Congestion Control in the EWSD Implementation

The implementation of the ACC in the system EWSD is detailed below (see Figure 2).

The EWSD implementation follows the ACC standard, ~~Congestion~~ in which congestion refers to CP congestion. At the congested VST, the degree of congestion is calculated with the aid of the STATOR algorithm, and the number and average processing time of all messages that have arrived at the CP, as well as the computing capacity, are taken into account. Potential congestion is converted into the overload priority level (OPL) with the congestion grades 0,1,2,3,4,5 and 6.

In the congested VST, the OPL is attached by the CP to commands with JC1=0 (job code 1) by the "piggy-back" method, respectively, and is thus routed to the line trunk groups LTGs. Commands with JC1=0 are SET-UP-COMPLETE (in the set-up of a call) and COME-AGAIN (request for more information). This method means that the OPL value on an LTG is only updated when a call is established via the LTG. In this way, there is a diffusion of the information, which has an adverse effect with respect to ACC. There are several exceptional rules in the method: At the beginning of an overload situation – i.e. when the STATOR first detects overload congestion following a phase with a normal load – an OPL of 3 is sent to all LTGs simultaneously via the command ADJUST-OPL. If the VTs input list (switching input list) at the CP is in danger of overrunning (more than 800 entries), then the OPL=6 is outputted with the command ADJUST-OPL.

At the LTG, the OPL is converted into an ACL in accordance with the following schema (the ACL value 0 stands for little or no congestion):

OPL	ACL
0	0
1	0
2	0
3	1
4	1
5	2
6	2

An LTG at the congested VST now sends an ACL to the neighbors with each message REL in accordance with Standard #7 (see Figure 3). There, the ACL is likewise received on an LTG. With the message REL-C (release complete), which informs the CP about the resolution ~~[sic]~~ of a call, the ACL is transported by the ~~GP~~ group processor (GP) of the LTG to the CP of the neighboring VST and is entered into the trunk data base ~~there~~ in bundles. This means that denial, or respectively, alternate routing takes place at the neighboring VST in bundles. Furthermore, this means that it is possible for different – that is, conflicting – information to be received nearly simultaneously at the neighboring VST to a bundle in the direction of the congested VST. ~~[sic]~~ This information can stem from different LTGs of the congested VST, for example. Positive ACL values are thus immediately overwritten.

In the trunk data base of the CP, a timer of 10s is wound up with every arrival of a positive ACL. When this timer expires without a new positive ACL having been received, the ACL in the trunk data base is decremented by 1, and the timer is rewound (in case of a new ACL value of 1).

At the neighboring exchanges, the type and intensity of the reaction is fixed in tables that are set up manually. The denial occurs globally as provided in the standard. Thus, for example, given an ACL of 1, 50% of all calls could be denied, and given an ACL of 2, 100%.

Three Main Problems of the ACC Algorithm

Primarily three weaknesses of the algorithm are responsible for the performance deficits in ACC that have been detected in simulations. All ~~lead it to heavy~~ of these lead abrupt vacillations in offered traffic, as a consequence of which lists ~~run to and over~~ exceed capacity, the queues grow long, and the throughput ~~slumps~~ declines.

Problem 1: Imprecise Control

The highly imprecise controlling of denial using only 2 ACL values and corresponding denial rates leads to heavy fluctuations in offered traffic, as a consequence of which the input lists at the CP grow very long, and the throughput ~~drops~~ declines.

Problem 2: "barn door effect"

When a VST overloads, it sends the same ACL to *all* neighboring VSTn (nearly) simultaneously. Thus, the same proportion of traffic at all neighboring VSTn across the board is choked off in the direction of the congested VST. Given an ACL of 1, i.e. "normal" congestion, this usually means that the congested VST is now offered less traffic than it can actually handle. It goes into *underload* , and the STATOR computes an OPL of 0. Despite this, the neighbors choke.off the traffic for at least 10s longer, until the timer has expired at all neighbors, the ACLs in the trunk data base have been set back there, and the *full* volume of traffic has been permitted. The "barn door" is then reopened. The VST immediately detects congestion again, and the process is repeated.

The protection period can last longer than the 10s of the timer if ACLs which have not been updated for a long time by a SET-UP-COMPLETE/COME-AGAIN command dwell at some LTGs of the congested VST. These incorrect ACLs are distributed to the neighbors via REL messages, where they overwrite zero values in the trunk data bases. Given heavy congestion, with ACL2, the effects of the timer are lost, since positive ACL values are immediately entered into the trunk data base. Yet the lingering non-updated ACLs produce a similarly unfavorable effect nevertheless.

The "barn door effect" thus likewise leads to heavy abrupt fluctuations of offered traffic. Long queues and wait times at the CP arise, and the throughput collapses.

Problem 3: Information deficits given high denial rates

The ACL is transferred in that it is attached to messages belonging to (SET-UP-COMPLETE, COME-AGAIN, REL, REL-C) calls. Thus, a certain proportion of calls must be successfully forwarded to the congested VST in order to guarantee the information exchange. But given high across-the -board denial rates (100%), situations arise in which almost no calls are routed to the congested VST. Too few SET-UP-COMPLETE commands leads to the OPL and ACL on the LTGs of the congested VST not being up to date. Too few REL messages results in obsolete ACLs on the neighboring exchanges. To prevent the endless persistence of overload values when the information exchange breaks down completely, the OPL on the LTGs of the congested VST is decremented every 4 seconds and is adapted to the ACL accordingly. This "emergency brake" can only ameliorate the ~~problemse [sic]~~ Problems, however.

If an attempt is made to solve these problems by shortening the 10s timer of a neighboring VST, the same negative effects appear. In addition, given heavy congestion, the protection phase is too short, and so the overloaded lists can no longer be cleared.

SUMMARY OF THE INVENTION

The invention is based on the object of solving the above described problems.

~~This object is achieved by a method as claimed in claim 1~~.

The invention improves protection control without an information exchange that has been established network-wide having to be modified or impaired between the VSTs ~~[sic]~~.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is detailed below with the aid of the drawing, which encompasses 4 Figures.

Figure 1 ~~shows~~ the ACC method under the ITU standard.

Figure 2 ~~shows~~ the determination of the degree of congestion (ACL value) in the congested VST.

Figure 3 ~~shows~~ the informing of a neighbor VST with the aid of delivered ACL values.

Figure 4 ~~shows~~ the evaluation of the ACL value information in the neighboring VST in bundles.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The inventive method computes what is known as an effective congestion value (OCL value) from the rough information (ACL values) received from a neighboring exchange, ~~which~~ value approximates the actual degree of congestion of a congested exchange, since ~~this comprises~~ a finer gradation, and ~~the values of the past~~ can be incorporated into its calculation. Unlike an actual momentary congestion grade, the calculated effective congestion value is smoothed in order to control the protection more softly (smoothing means that peaks in the characteristic curve of the actual congestion value are damped). The protection regimens thus fluctuate less extremely.

The simulations demonstrate very good performance data (calls put through, wait times, queue lengths, processor usage) at the congested exchanges particularly in conjunction with another denial method (denial at the congested exchange itself in accordance with the "overload priority level OPL"), which is used at Siemens exchanges. But the invention meaningfully relieves the exchanges without additional protection measures besides ACC.

Recovery of the actual level of congestion and smoothing

The following exemplifying embodiments are all based on the idea of recovering the actual level of congestion from the rough ACL values 1 and 2 and the indirect information from an REL message without ACL (meaning no congestion) and smoothing this in order to stabilize the protection measures. To this end, an effective degree of congestion, known as the OCL (overload congestion level), is determined from the history of the received ACLs, that is, from the information of several past ACLs, and

this is mapped onto a protection control value in accordance with which the neighboring exchanges protect in a finer gradation. The finer gradation can be laid out as ~~conformal~~ conforming to the ITU standard (see exemplifying embodiments).

Exemplifying embodiments for calculating the smoothed overload congestion level OCL are given below.

Example 1: Calculation of the overload congestion level from all ACLs that are received within a ceratin time frame

Though an individual ACL value gives only very approximate information about the overload situation, one can obtain more precise statements, which are also smoothed, by considering the values which arrived previously and determining a reasonable average value. The following approach was tested in simulations and leads to significant improvements of the performance:

At an interval of one second, respectively, the average value over all ACLs that arrived in the preceding second interval is formed. REL messages that arrive without ACC information are assigned the pseudo-ACL-value 0. In this way, the negative information of an empty REL message is also used; that is, the information "there is no overload". A(j) characterizes the average value over the ACLs and pseudo-ACLs for the interval which began j seconds ago. The average values are weighted and added, and thus the new overload congestion level OCL is formed as a weighted average. The OCL is a compressed image of the OPL and can accept all values between 0 and 2.

$$OCL = \sum_{j=1,\ldots,n} w(j) \cdot A(j), \quad n = 20, \quad w(j) = \frac{1/\sqrt{j}}{\sum_{k=1}^{20} 1/\sqrt{k}}$$

Formula 1: OCL determination (each second) from the ACL average values of the last 20 one-second intervals.

Good results were obtained in simulations with the above described weights *w(j)*. Of course other weights are possible in principle, ~~whereby~~ but for selected weightings it is important for the smoothing that the weights do not drop too rapidly as the index *j* rises.

As another possibility, the OCL can be determined semi-~~recusively [sic]~~ recursively. This presupposes an initialization of the OCL. The first ACL average value is used for this.

$$OCL_{new} = \alpha \cdot OCL_{old} + (1-\alpha) \cdot A(1)$$

Formula 2: semi-recursive OCL determination

Selecting $\alpha=1/2$, one obtains the terms of the geometric series as weights. With $\alpha=0.9$, one obtains a similarly good functioning of the method in simulations as when the sum formula in formula 1 is used. With the aid of simulations, the optimal value for α can be determined.

Protection occurs in accordance with the following schema:

Congestion Level	Protection in %
0.00-0.249	0
0.25-0.499	12.5
0.50-0.749	25
0.75-0.999	37.5
1.00-1.249	50
1.25-1.499	62.5
1.50-1.749	75
1.75-1.999	87.5
2.00	100

Table 2: Protection according to OCL values

According to table 2, each OCL is mapped into a value for controlling the protection level, which is referred to ~~henceforth~~ below as a protection control value.

Example 2: calculation of a smoothed OCLs with the aid of the timer

Here, the 8 gradations of the OPL are provided for the OCL: 0, 1, 2, 3, 4, 5, 6, 7, 8. In addition there are two basic statuses of a (neighboring) VST with respect to congestion:

– "no congestion detected",

– "congestion detected".

When an ACL>0 is received for the first time following a phase of normal load, the basic status changes to congestion detected. The OCL is then initialized, for instance to the value 4. At the same time, a timer of 1 second is started. When the timer runs out, the average value A(1) over all ACLs in the preceding one-second interval is used to adjust the OCL:

$$A(1) \geq \alpha \Rightarrow OCL_{new} = \min\left(OCL_{old} + 1.8\right),$$
$$A(1) \leq \beta \Rightarrow OCL_{new} = \max\left(OCL_{old} - 1.0\right).$$

Formula 3: OCL adjustment upon timer expiration

For α and ß the values α=1.5 and ß=0.5 are suggested. The ~~optical~~ optimal parameters can be found via simulations. Past simulations have demonstrated that the method would have to be refined – perhaps by introducing additional threshold values – in order to achieve a performance capability comparable to the method described ~~[...]~~ in the previous section.

If the OCL remains at 0 in a 10-second interval (a second timer), and only ACL=0 is received, the VST reverts to the basic status "no congestion detected". Choke-off occurs as in the above proposal.

Congestion Level	Protection in %
0	0
1	12.5
2	25
3	37.5
4	50
5	62.5
6	75
7	87.5
8	100

Table 3: Protection according to OCL values

Calculation upon consideration of the frequency of incoming ACLs

Another possibility for recovering information is to integrate the frequency of incoming ACLs: The OCL is updated upon reception of each ACL. The OCL values 0,1,2,3,4,5,6,7,8 are again used, and protection follows table 3. The adjustment of the OCL is controlled as follows (initialization of the OCL_alt: OCL_alt=0).

OCL_alt	ACL=0	ACL=1	ACL=2
0,1,2	$OCL_{new}=\max(OCL_{old}-1.0)$	$OCL_{new}=\min(OCL_{old}+1.8)$	$OCL_{new}=\min(OCL_{old}+2.8)$
3,4	$OCL_{new}=\max(OCL_{old}-1.0)$	$OCL_{new}=\min(OCL_{old}+1.8)$	$OCL_{new}=\min(OCL_{old}+2.8)$
5,6	$OCL_{new}=\max(OCL_{old}-1.0)$	$OCL_{new}=OCL_{old}$	$OCL_{new}=\min(OCL_{old}+1.8)$
7,8	$OCL_{new}=\max(OCL_{old}-2.0)$	$OCL_{new}=\max(OCL_{old}-1.0)$	$OCL_{new}=\min(OCL_{old}+1.8)$

Formula 4: Adjustment of the OPL with each incoming ACL (accounting for frequency)

Further smoothing could be achieved in that all ACLs coming in in a certain time interval after an adjustment of the OCL are ignored. However, such a smoothing concept would conflict with the desire to exploit incoming positive ACL values.

Ins A7

Abbreviations

ACC	automatic congestion control
ACL	automatic congestion level
CP	coordination processor
GP	group processor – processor on the LTG
LTP	line trunk group
OCL	overload congestion level
OPL	overload priority level
REL	order: release
REL-C	message: release complete
VST	exchange